

EXTENDING WORD PROBLEMS IN DETERMINISTIC FINITE AUTOMATA

MIKHAIL V. BERLINKOV

*Institute of Mathematics and Computer Science,
Ural Federal University, Ekaterinburg, Russia*

ROBERT FERENS

*Institute of Computer Science,
University of Wrocław, Wrocław, Poland*

MAREK SZYKUŁA

*Institute of Computer Science,
University of Wrocław, Wrocław, Poland*

ABSTRACT. A word w is *extending* a subset of states S of a deterministic finite automaton, if the set of states mapped to S by w (the preimage of S under the action of w) is larger than S . This notion together with its variations has particular importance in the field of synchronizing automata, where a number of methods and algorithms rely on finding (short) extending words. In this paper we study the complexity of several variants of extending word problems: deciding whether there exists an extending word, an extending word that extends to the whole set of states, a word avoiding a state, and a word that either extends or shrinks the subset. Additionally, we study the complexity of these problems when an upper bound on the length of the word is also given, and we consider the subclasses of strongly connected, synchronizing, binary, and unary automata. We show either hardness or polynomial algorithms for the considered variants.

KEYWORDS: avoiding word, extending word, extensible subset, reset word, synchronizing automaton

1. INTRODUCTION

A deterministic finite complete (semi)automaton \mathcal{A} is a triple (Q, Σ, δ) , where Q is the set of *states*, Σ is the input *alphabet*, and $\delta: Q \times \Sigma \rightarrow Q$ is the *transition function*. We extend δ to a function $Q \times \Sigma^* \rightarrow Q$ in the usual way. Throughout the paper, by n we always denote the number of states $|Q|$.

E-mail addresses: berlm@mail.ru, robert.ferens@interia.pl, msz@cs.uni.wroc.pl.

When the context automaton is clear, given a state $q \in Q$ and a word $w \in \Sigma^*$ we write shortly $q \cdot w = \delta(q, w)$. Given a subset $S \subseteq Q$, the *image* of S under the action of a word $w \in \Sigma^*$ is $S \cdot w = \delta(S, w) = \{q \cdot w \mid q \in S\}$. The *preimage* is $S \cdot w^{-1} = \delta^{-1}(S, w) = \{q \in Q \mid q \cdot w \in S\}$.

We say that a word w *compresses* a subset S if $|S \cdot w| < |S|$, and it *extends* S if $|S \cdot w^{-1}| > |S|$. We also say that a subset S is *compressible* if there is a word that compresses it, and it is *extendible* if there is a word that extends it. The *rank* of a word w is the cardinality of the image $Q \cdot w$. A word of rank 1 is called *reset* or *synchronizing*, and an automaton that admits a reset word is called *synchronizing*. Also, for a subset $S \subseteq Q$, we say that a word $w \in \Sigma^*$ such that $|S \cdot w| = 1$ *synchronizes* S .

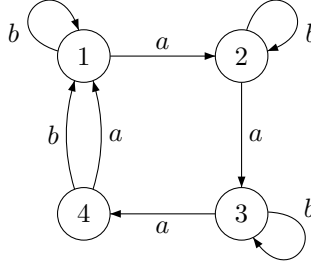


FIGURE 1. The Černý automaton with 4 states.

Fig. 1 shows an example. For $S = \{2, 3\}$, the shortest compressing word is aab , and we have $\{2, 3\} \cdot aab = \{1\}$, while the shortest extending word is ba , and we have $\{2, 3\} \cdot (ba)^{-1} = \{1, 2\} \cdot b^{-1} = \{1, 2, 4\}$.

Synchronizing automata serve as transparent and natural models of various systems in many applications in different fields, such as coding theory, DNA-computing, robotics, testing of reactive systems, and theory of information sources. They also reveal interesting connections with symbolic dynamics, language theory, and many other parts of mathematics. For a detailed introduction to the theory of synchronizing automata we refer the reader to the surveys [20, 35], and for a review of relations with coding theory to [18].

The famous Černý conjecture ([12]), stated in 1969, is one of the most longstanding open problems in automata theory, and is the central problem in the theory of synchronizing automata. It states that a synchronizing automaton has a reset word of length at most $(n - 1)^2$.

Words compressing and extending subsets play a crucial role in synchronizing automata. In fact, all known algorithms (e.g. [1, 9, 13, 21]) and proofs of upper bounds on the length of the shortest reset words use finding a word that compresses a subset (e.g. [2, 9, 10, 13, 16, 33, 36]) and/or a word that extends a subset (e.g. [3, 4, 9, 29]). See [20, 27, 35] for surveys.

1.1. Compressing a subset. The complexity of problems related to compressing a subset has been well studied.

It is known that, given an automaton \mathcal{A} and a subset $S \subseteq Q$, determining whether there is a word that synchronizes it is PSPACE-complete [26]. The same holds even for strongly connected binary automata [37].

On the other hand, checking whether the automaton is synchronizing (whether there is a word that synchronizes Q) can be solved in $O(kn^2)$ time and space [12, 13, 35] and in $O(n)$ average time and space for the random binary case [6]. To this end, we just verify whether all pairs of

states are compressible. Using the same algorithm, we can determine whether a given subset is compressible. From this fact, it is easy to see that these problems are NL-complete (not explicitly stated in literature). However, it remains unknown and seems difficult to show whether it remains NL-complete when the automaton is strongly connected.

Deciding whether there exists a synchronizing word of a given length is NP-complete [13] (cf. [24] for the complexity of the corresponding functional problems), even if the given automaton is binary. We additionally state here that the problem remains NP-complete when the automaton is strongly connected and binary. However, deciding whether there exists a word that only compresses a subset still can be solved in $O(|\Sigma|n^2)$ time, as for every pair of states we can compute a shortest word that compresses the pair. The unary case is trivial, as the automaton is synchronizing if and only if the transformation of the letter forms a directed tree.

We refer also to [23, 26] for the cases of NFA and PDFA (*partial deterministic finite automata*), to [17] for partial observability, and to [11] for reachability of a subset.

1.2. Extending a subset and our contributions. In contrast to the problems related to images (compression), the complexity of the problems related to preimages was not well understood.

In the paper we study computational complexity of problems of determining extending words in various cases. We focus on the decision problems of existence of an extending word and whether the length of the shortest extending words is at most the given bound. We also consider the subclasses of strongly connected and synchronizing automata, which are particularly important in the area.

- (1) In Section 2 we study the problem of existence of a word whose inverse action results in a larger preimage of the given subset (Problem 1 and 3). This is the basic step of the so-called *extension method* of finding a reset word. Also, we analyze the problem of extending the subset to the whole set of states (Problem 2 and 4), which, can be viewed as a generalization of finding a reset word. We show that these problems are computationally difficult (PSPACE-complete or NP-complete) in most cases.
- (2) Section 3 is devoted to *avoiding words* (Problem 5 and 6). A word is avoiding a state if no other state is mapped to this state under the action of this word. The concept of avoiding appeared recently, and is related with an attempt to improve the general upper bound on the length of the shortest reset words [15, 31, 34]. It can be seen they are a particular case of extending words, namely, an avoiding word can be viewed as a word extending a subset of size $n - 1$ to Q . Here we characterize the states that are avoidable and provide a polynomial algorithm finding an avoiding word. Finding a shortest avoiding word remains computationally difficult (NP-complete).
- (3) In Section 4 we consider the problem of determining a word that results in a subset having a different size (Problem 7 and 8). Interestingly, in contrast with the computationally difficult problems of finding a word that extends the subset and finding a word that shrinks the subset, for this variant there exists a polynomial algorithm, which uses linear algebra applied for automata.

Table 1 and Table 2 summarize our results together with known results about compressing. For the cases where a polynomial algorithm exists, we put the time complexity of the best one. All the results for non-unary automata remain the same also in the case of a binary alphabet.

We also state a few open problems that were not solved in this paper. Another set of open problems contains questions about the bounds on the length of a shortest extending word. From the complexity results we may infer certain exponential or polynomial bounds, though not precise

TABLE 1. Computational complexity of decision problems in classes of automata:
Is there a word $w \in \Sigma^*$ such that:

Subclass of automata	All and strongly connected	Synchronizing	Str. con. and synch.	Unary
$ S \cdot w = 1$	PSPACE-complete [26, 37]	$O(1)$	$O(1)$	$O(n)$
$ S \cdot w < S $	$O(\Sigma n^2)$ [12, 35]	$O(1)$	$O(1)$	$O(n)$
(Problem 1) $ S \cdot w^{-1} > S $	PSPACE-complete (Thm. 2)	PSPACE-complete (Prop. 3)	$O(1)$	NP-complete (Thm. 7)
(Problem 2) $S \cdot w^{-1} = Q$	PSPACE-complete (Thm. 2)	$O(\Sigma n)$ (Thm. 5)	$O(1)$	$O(n)$ (Prop. 8)
(Problem 5) $(Q \setminus \{q\}) \cdot w^{-1} = Q$	$O(\Sigma n^2)$ (Thm. 10)	$O(1)$	$O(1)$	$O(n)$ (Prop. 14)
(Problem. 7) $ S \cdot w^{-1} \neq S $	$O(\Sigma n^3)$ (Thm. 16)	$O(1)$	$O(1)$	$O(n^2)$ (Prop. 17)

bounds. In [8] it was shown that in the case of strongly connected synchronizing binary automata, the length of the shortest extending words can be quadratic, which would be best possible if the Černý conjecture is true.

2. EXTENDING A SUBSET

We deal with the following problems:

Problem 1. *Given $\mathcal{A} = (Q, \Sigma, \delta)$ and a subset $S \subseteq Q$, is S extensible?*

Problem 2. *Given $\mathcal{A} = (Q, \Sigma, \delta)$ and a subset $S \subseteq Q$, is there a word $w \in \Sigma^*$ such that $S \cdot w^{-1} = Q$?*

We note that the cases in which the size of S is decreased (becomes zero, respectively) under the inverse action are equivalent.

Remark 1. $|S \cdot w^{-1}| > |S|$ if and only if $|(Q \setminus S) \cdot w^{-1}| < |Q \setminus S|$.

Theorem 2. *Problem 1 and Problem 2 are PSPACE-complete even if \mathcal{A} is strongly connected.*

Proof. To solve one of the problems in NPSpace, we guess the length of a word w with the required property, and then guess the letters of w from the end. Of course we do not store w , which may have exponential length, but just keep the subset $S \cdot u^{-1}$, where u is the current suffix of w . The current subset can be stored in $O(n)$, and since there are 2^n different subsets, $|w| \leq 2^n$ and the current length also can be stored in $O(n)$. Therefore, by Savitch's theorem, the problems are in PSPACE.

To show PSPACE-hardness, we are going to reduce from the problem of determining whether an intersection of regular languages given as DFAs is non-empty.

TABLE 2. Complexity of decision problems: Is there are a word $w \in \Sigma^*$ of length $\leq \ell$ such that:

Subclass of automata	All and strongly connected	Synchronizing	Str. con. and synch.	Unary
$ S \cdot w = 1$	PSPACE-complete [26, 37]	NP-complete [13]	NP-complete (Cor. 13)	$O(n)$
$ S \cdot w < S $	$O(\Sigma n^2)$ [13]	$O(\Sigma n^2)$ [13]	$O(\Sigma n^2)$ [13]	$O(n)$
(Problem 3) $ S \cdot w^{-1} > S $	PSPACE-complete (Thm. 2)	PSPACE-complete (Prop. 3)	NP-complete (Thm. 11)	NP-complete (Thm. 7)
(Problem 4) $S \cdot w^{-1} = Q$	PSPACE-complete (Thm. 2)	NP-complete (Cor. 12)	NP-complete (Cor. 12)	$O(n)$ (Prop. 8)
(Problem 6) $(Q \setminus \{q\})w^{-1} = Q$	NP-complete (Thm. 11)	NP-complete (Thm. 11)	NP-complete (Thm. 11)	$O(n)$ (Prop. 14)
(Problem 8) $ S \cdot w^{-1} \neq S $	$O(\Sigma n^3)$ (Thm. 16)	$O(\Sigma n^3)$ (Thm. 16)	$O(\Sigma n^3)$ (Thm. 16)	$O(n^2)$ (Prop. 17)

Let $(\mathcal{D}_i)_{i \in \{1, \dots, m\}}$ be the given sequence of DFAs with an i -th automaton $\mathcal{D}_i = (Q_i, \Sigma, \delta_i, s_i, F_i)$ recognizing a language \mathcal{L}_i , where Q_i is the set of states, Σ is the common alphabet, δ_i is the transition function, s_i is the initial state, and F_i is the set of final states. The problem if there exists a word accepted by all $\mathcal{D}_1, \dots, \mathcal{D}_m$ (the intersection of L_i is non-empty) is a well known PSPACE-complete problem, called Finite Automata Intersection [22].

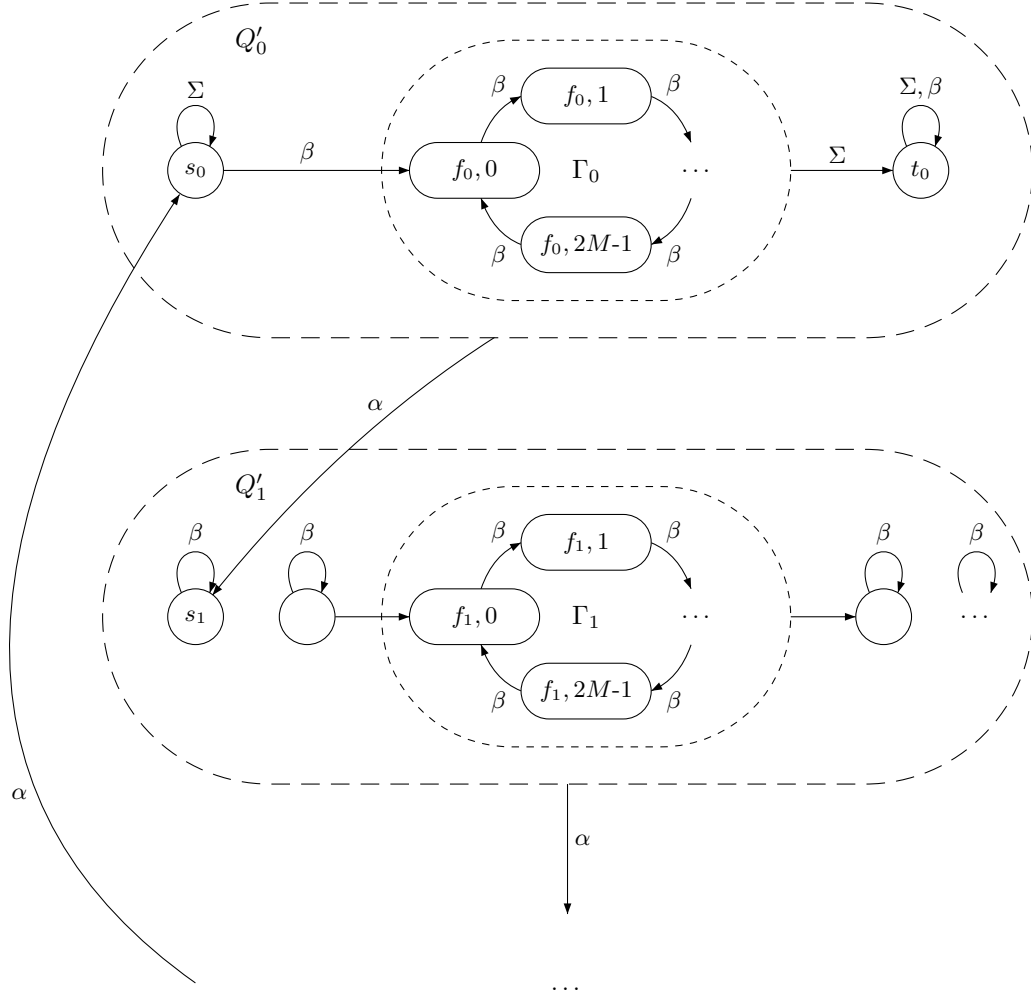
For each \mathcal{D}_i we choose an arbitrary $f_i \in F_i$. Let $M = \sum_{i=1}^m |Q_i|$. We construct the semiautomaton $\mathcal{D} = (Q', \Sigma', \delta')$, and define $S \subseteq Q'$ as an instance of our both problems. The scheme of the automaton is shown in Fig. 2.

- Let $\Gamma_i = \{f_i\} \times \{0, \dots, 2M - 1\}$ be fresh states. Let $Q'_i = (Q_i \setminus \{f_i\}) \cup \Gamma_i$. Let $Q'_0 = \{s_0, t_0\} \cup \Gamma_0$, where s_0, t_0 . Then $Q' = \bigcup_{i=0}^m Q'_i$.
- $\Sigma' = \Sigma \cup \{\alpha, \beta\}$, where α and β are fresh letters.
- δ' is defined by:
 - For $q \in Q_i \setminus \{f_i\}$ and $a \in \Sigma$, we have

$$\delta'(q, a) = \begin{cases} \delta_i(q, a) & \text{if } \delta_i(q, a) \neq f_i, \\ (f_i, 0) & \text{otherwise.} \end{cases}$$

- For $a \in \Sigma$, we have

$$\delta'(t_0, a) = t_0, \quad \delta'(s_0, a) = s_0.$$

FIGURE 2. The automaton \mathcal{D} from the proof of Theorem 2.

- For $k \in \{0, \dots, 2M - 1\}$, $i \in \{1, \dots, m\}$, and $a \in \Sigma$, we have

$$\begin{aligned} \delta'((f_0, k), a) &= t_0, \\ \delta'((f_i, k), a) &= \begin{cases} \delta_i(f_i, a) & \text{if } \delta_i(q, a) \neq f_i, \\ (f_i, 0) & \text{otherwise.} \end{cases} \end{aligned}$$

- For $q \in Q'_i$, we have

$$\delta'(q, \alpha) = s_{(i+1) \bmod (m+1)}.$$

- For $i \in \{0, \dots, m\}$ and $k \in \{0, \dots, 2M - 1\}$, we have

$$\delta'((f_i, k), \beta) = (f_i, k + 1 \bmod 2M).$$

– We have

$$\delta'(s_0, \beta) = (f_0, 0).$$

– For the remaining states $q \in Q' \setminus (\bigcup_{i=0}^m \Gamma_i \cup \{s_0\})$, we have

$$\delta'(q, \beta) = q.$$

• The subset $S \subseteq Q'$ is

$$S = \left(\bigcup_{i=1}^m F_i \cap Q' \right) \cup \bigcup_{i=0}^m \Gamma_i \cup \{s_0\}.$$

It is easy to observe that \mathcal{D} is strongly connected. Get any $i, j \in \{0, \dots, m\}$. We show how to reach any state $q \in Q'_j$ from a state $p \in Q'_i$. First, we can reach s_j by $\alpha^{(m+1+j-i) \bmod (m+1)}$. For $j \geq 1$, each state $q \in Q'_j \setminus (\Gamma_j \setminus \{(f_j, 0)\})$ is reachable from s_j , since δ' restricted to Σ acts on Q'_j as δ_j on Q_j (with f_j replaced by $(f_j, 0)$) and \mathcal{D}_j is minimal. For $j = 0$, states $(f_0, 0)$ and t_0 are reachable from s_0 by the transformations of β and βa respectively, for any $a \in \Sigma$. States $q \in \Gamma_j$ can be reached from $(f_j, 0)$ using δ_β .

We will show the following statements:

- (1) If S is extensible in \mathcal{D} , then the intersection of the languages L_i is non-empty.
- (2) If the intersection of the languages L_i is non-empty, then S is extensible to Q in \mathcal{D} .

This will prove that the intersection of the languages L_i is non-empty if and only if S is extensible, which is also equivalent to that S is extensible to Q .

(1): Observe that, for each $i \in \{0, \dots, m\}$, if $(S \cdot w^{-1}) \cap \Gamma_i \neq \emptyset$, then $(S \cdot w^{-1}) \cap \Gamma_i = \Gamma_i$. This follows by induction since: the empty word possesses this property; the transformation δ_a for $a \in \Sigma \setminus \{\beta\}$ maps every state from Γ_i to the same state, so it preserves the property; δ_β acts cyclically on Γ_i so also preserves the property.

Suppose that S is extensible by a word w . Notice that, M is an upper bound on the number of states in $Q' \setminus \bigcup_{i=0}^m \Gamma_i$ (for $m \geq 2$). We also have $|S| \geq 1 + (m+1) \cdot 2M$. We conclude that $\Gamma_i \subseteq S \cdot w^{-1}$ for each $i \in \{0, \dots, m\}$, since

$$|Q' \setminus \Gamma_i| \leq m \cdot 2M + M \leq (m+1) \cdot 2M < |S|,$$

so $(S \cdot w^{-1}) \cap \Gamma_i \neq \emptyset$ and then our previous observation $\Gamma_i \subseteq S \cdot w^{-1}$.

Now, the extending word w must contain the letter α . For a contradiction, if $w \in (\Sigma' \setminus \{\alpha\})^*$, then if it contains a letter $a \in \Sigma$, then $S \cdot w^{-1}$ does not contain any state from $\Gamma_0 \cup \{t_0\}$, as the only outgoing edges from this subset are labeled by α , $t_0 \notin S$, $\Gamma_0 \cdot \beta^{-1} = \Gamma_0$, and $\Gamma_0 \cdot a^{-1} = \emptyset$. This contradicts the previous paragraph. Also, w cannot be of the form β^k , for $k \in \mathbb{N}$, since $S \cdot \beta^k = S$. Hence, $w = w_p \alpha w_s$, where $w_p \in (\Sigma')^*$ and $w_s \in (\Sigma' \setminus \{\alpha\})^*$.

Note that if T is a subset of Q such that $T \cap Q_i = \emptyset$ for some i , then also $(T \cdot u^{-1}) \cap Q_{i'} = \emptyset$ for every word u and some i' ; because only α maps states Q_i outside Q_i , and it acts cyclically on these sets. Hence, in this case, every preimage of T does not contain some $\Gamma_{i'}$ set. So $\{s_i \mid i \in \{0, \dots, m\}\} \subseteq S \cdot (w_s)^{-1}$, since in the opposite case $(S \cdot (\alpha w_s)^{-1}) \cap Q_i = \emptyset$ for some i .

Let w'_s be the word obtained by removing all β letters from w_s . Note that, for every $i \in \{1, \dots, m\}$ and every suffix u of w_s , we have $(S \cdot u^{-1}) \cap Q'_i = (S \cdot (\beta u)^{-1}) \cap Q'_i$. Hence, $(S \cdot w_s^{-1}) \cap (Q' \setminus Q'_0) = S \cdot (w'_s)^{-1} \cap (Q' \setminus Q'_0)$.

Now, the word w'_s is in Σ^* , and $S \cdot w_s^{-1}$ contains s_i for all $i \in \{1, \dots, m\}$. Hence, the action of w'_s maps s_i to either a state in $F_i \setminus \{f_i\}$ or $(f_i, 0)$, which means that w'_s maps s_i to F_i in \mathcal{D}_i . Therefore, w'_s is in the intersection of the languages L_i .

(2): Suppose that the intersection of the languages L_i is non-empty, so there exists a word $w \in \Sigma^*$ such that $s_i \cdot w \in F_i$ for every i . Then we have $S \cdot (\alpha w)^{-1} = Q$, thus S is extensible to Q . \square

Proposition 3. *When the automaton is synchronizing, Problem 1 remains PSPACE-complete.*

Proof. We just add a sink state z and a letter whose synchronizes $\mathcal{A}(Q, \Sigma, \delta)$ to z .

Formally, we construct $\mathcal{A}'(Q \cup \{z\}, \Sigma \cup \{a_z\}, \delta')$, where

$$\delta'(q, a) = \begin{cases} \delta(q, a) & \text{if } q \in Q \text{ and } a \in \Sigma; \\ z & \text{if } q = z \text{ or } a = a_z. \end{cases}$$

Then \mathcal{A}' is synchronizing and S is extensible in \mathcal{A}' if and only if it is extensible in \mathcal{A} , since applying a_z for any preimage $S \cdot w^{-1}$ always results in \emptyset . \square

Note that in the case of strongly connected synchronizing automaton, both problems have a trivial solution, since every non-empty proper subset of Q is extensible to Q (by a suitable reset word).

It remains to ensure that the problems remain hard in the case of a binary alphabet:

Theorem 4. *Given an automaton $\mathcal{A}(Q, \Sigma, \delta)$ and a subset $S \subseteq Q$, we can construct in polynomial time a binary automaton $\mathcal{A}'(Q', \{a', b'\}, \delta')$ such that*

(1) \mathcal{A} is strongly connected if and only if \mathcal{A}' is strongly connected,

and a subsets $S', T' \subseteq Q'$ such that either:

(2) S' is extensible in \mathcal{A}' if and only if S is extensible in \mathcal{A} , or

(3) T' is extensible to Q in \mathcal{A}' if and only if S is extensible to Q in \mathcal{A} .

Proof. We can assume that $\Sigma = \{a_0, \dots, a_{k-1}\}$. We construct $\mathcal{A}' = (Q', \{a', b'\}, \delta')$ with $Q' = Q \times \Sigma$ and δ' defined as follows: $\delta'((q, a_i), a') = (\delta(q, a_i), a_i)$, and $\delta'((q, a_i), b') = (q, a_{(i+1) \bmod k})$. Clearly, \mathcal{A}' can be constructed in $O(nk)$ time, where $k = |\Sigma|$.

(1): Suppose that \mathcal{A} is strongly connected; we will show that \mathcal{A}' is also strongly connected. Let (q_1, a_i) and (q_2, a_j) be any two states of \mathcal{A}' . In \mathcal{A} , there is a word w such that $q_1 \cdot w = q_2$. Let w' be the word obtained from w by replacing every letter a_h by the word $(b')^h a' (b')^{k-h}$. Note that in \mathcal{A}' we have

$$(p, a_0) \cdot (b')^h a' (b')^{k-h} = (p \cdot a_h, a_0),$$

hence $(q_1, a_0) \cdot w' = (q_1 \cdot w, a_0)$. Then the action of the word $(b')^{k-i} w' (b')^j$ maps (q_1, a_i) to (q_2, a_j) .

Conversely, suppose that \mathcal{A}' is strongly connected, so every (q_1, a_i) can be mapped to every (q_2, a_j) by the action of a word w' . Then

$$w' = (b')^{h_1} a' \dots (b')^{h_{m-1}} a' (b')^{h_m},$$

for some $m \geq 1$ and $h_1, \dots, h_m \geq 0$. We construct w of length $m-1$, where the s -th letter is a_r with $r = (i + \sum_{j=1}^s h_j) \bmod k$. Then w maps q_1 to q_2 in \mathcal{A} .

(2) and (3): For $i \in \{0, \dots, k-1\}$ we define $U_i = (Q \times \{\Sigma \setminus \{a_i\}\})$. Observe that for any word $u' \in \{a', b'\}^*$, we have $U_i \cdot (u')^{-1} = U_j$ for some j , which depends on i and the number of letters b' in u' .

We define

$$S' = (S \times \{a_0\}) \cup U_0.$$

Suppose that S is extensible in \mathcal{A} by a word w , and let w' be the word obtained from w as in (1). Then $(w')^{-1}$ maps U_0 to U_0 , and $(S \times \{a_0\})$ to $(S \cdot w^{-1}) \times \{a_0\}$. We have:

$$S'(w')^{-1} = ((S \cdot w^{-1}) \times \{a_0\}) \cup U_0,$$

and since $|S \cdot w^{-1}| > |S|$, this means that w' extends S' . By the same argument, if w extends S to Q , then w' extends S' to Q' .

Conversely, suppose that S' is extensible in \mathcal{A}' by a word w' , and let w be the word obtained from w' as in (1). Then, for some i , we have

$$S' \cdot (w')^{-1} = ((S \cdot w^{-1}) \times \{a_i\}) \cup U_i,$$

and since $|U_0| = |U_i|$ it must be that $|S \cdot w^{-1}| > |S|$. Also, if $S' \cdot (w')^{-1} = Q'$ then $S \cdot w^{-1} = Q$. \square

Proposition 5. *When the automaton is synchronizing, Problem 2 can be solved in $O(|\Sigma|n)$ time.*

Proof. Since \mathcal{A} is synchronizing, Problem 2 reduces to checking whether there is $q \in S$ reachable from every state: It is well known that a synchronizing automaton has precisely one strongly connected *sink* component that is reachable from every state. If w is a reset word that synchronizes Q to p , and u is such that $p \cdot q$, then wu extends $\{q\}$ to Q . If S does not contain a state from the sink component, then any preimage of S also does not contain these states.

The problem can be solved in $O(n \cdot |\Sigma|)$, since the states of the sink component can be determined in linear time by Tarjan's algorithm [32]. \square

We also determine the complexity class:

Theorem 6. *When the automaton is synchronizing, problem 2 is NL-complete.*

Proof. It is easy to see that the problem is in NL. We guess a state $q \in S$ from the sink component and verify in logarithmic space that it is reachable from every state.

For NL-hardness, we reduce from ST-connectivity: Given a graph $G(V, E)$ and vertices s, t , check whether there is a path from s to t . We will output a strongly connected synchronizing automaton $\mathcal{A}(V, \Sigma, \delta)$ and $S \subseteq Q$ such that S is extensible to Q if and only if there is a path from s to t in G .

First, we compute the maximum output degree of G , and set $\Sigma = \Sigma' \cup \{\alpha\}$, where $|\Sigma'|$ is equal to the maximum output degree. We output \mathcal{A} such that for every $q \in V$, every edge $(q, p) \in E$ is colored by a different letter from Σ' . If there is no outgoing edge from q , then we set the transitions of all letters from Σ' to be loops. If the output degree is smaller than $|\Sigma'|$, then we simply repeat the transition of the last letter. Next, we define $\delta(q, \alpha) = s$ for every $q \in V$. Finally, let $S = \{t\}$. The reduction uses logarithmic space, since it requires only counting and enumerating through V and Σ' . The produced automaton \mathcal{A} is synchronizing just by α .

Suppose that there is a path from s to t . Then there is a word w such that $\delta(s, w) = t$, and so $\{t\} \cdot (\alpha w)^{-1} = Q$.

Suppose that $\{t\}$ is extensible to Q by some word w . Let w' be the longest suffix of w that does not contain α . Since α^{-1} results in \emptyset for any subset not containing s , it must be that $s \in \{t\}(w')^{-1}$. Hence $\delta(s, w') = t$, and the path labeled by w' is the path from s to t in G . \square

2.1. Bounded length. We turn our attention to the variants in which an upper bound on the length of word w is also given.

Problem 3. *Given $\mathcal{A} = (Q, \Sigma, \delta)$, a subset $S \subseteq Q$, and a number ℓ , is S extensible by a word of length at most ℓ ?*

Problem 4. Given $\mathcal{A} = (Q, \Sigma, \delta)$, a subset $S \subseteq Q$, and a number ℓ , is there a word $w \in \Sigma^*$ such that $S \cdot w^{-1} = Q$ of length at most ℓ ?

Obviously, both these problems remain PSPACE-complete in all cases (also when the automaton is binary), as we can set $\ell = 2^n$. When the automaton is synchronizing, Problem 4 is NP-complete, which will be shown in Section 3.

2.2. Unary case. In contrast to the binary case, in the unary case the complexity of both problems is distinct. Determining whether a subset is extensible is NP-complete, while extending to Q can be solved in linear time.

Theorem 7. *When the automaton is unary, Problem 1 is NP-complete.*

Proof. We show that the problem is in NP. First we non-deterministically guess $\ell \leq 2^n$. We observe that $([a]^T)^\ell [S] = [S \cdot (a^{-1})^\ell]$. The ℓ -th power of $[a]^T$ can be computed in $O(\log \ell) = O(n)$ time by the usual doubling algorithm for fast matrix exponentiation. Then we simply verify if the resulted vector $([a]^T)^\ell [S]$ has more 1 than $[S]$.

For NP-hardness, we use a similar idea to that from [30] for coNP-hardness of universality of the language accepted by a unary NFA.

By p_i we denote the i -th prime number. Given an instance φ of 3-SAT, where \mathcal{V} is the set of variables and \mathcal{C} is the set of clauses, we construct a unary automaton $\mathcal{A}(Q, \{a\}, \delta)$ as follows. For every clause $C \in \mathcal{C}$ with variables v_x, v_y , and v_z , we create a cycle in the automaton of length $t = p_x p_y p_z$. Let q_0, \dots, q_{t-1} be the states of the cycle in the reverse order ($\delta(q_{(i+1) \bmod t}, a) = q_i$, for $i = 1, \dots, t$). For every i such that:

- (1) $p_x \mid i \iff v_x$ occurs positively in C , and
- (2) $p_y \mid i \iff v_y$ occurs positively in C , and
- (3) $p_z \mid i \iff v_z$ occurs positively in C ,

we add a new state p_i and the transition $\delta(p_i, a) = q_{(i-1) \bmod t}$. Next, we add $|\mathcal{C}| - 1$ states r_i , a state r , and the transitions $\delta(r_i, a) = r$ for each i , and $\delta(r, a) = r$. Finally, we let S to contain the state q_0 from each cycle, and all states r_i .

We claim that φ is satisfiable if and only if S is extensible. First, suppose that S is extensible by $(a^{-1})^\ell$. We construct the assignment I such that an i -th variable v_i is true if and only if $p_i \mid \ell$. Clearly, $S \cdot (a^{-1})^\ell$ for every $\ell > 0$ does not contain states r_i nor r . Moreover, $S \cdot (a^{-1})^\ell$ contains exactly one state from each cycle, and possibly one state p_i for a cycle. Since S has size $2|\mathcal{C}| - 1$, the extended subset $S \cdot (a^{-1})^\ell$ must have size at least $2|\mathcal{C}|$ and so contain p_i for every cycle. Consider one of the cycles and the corresponding clause. Note that $S \cdot (a^{-1})^\ell$ contains $q_{j \bmod t}$ and it can contain only $p_{\ell \bmod t}$ if it exists. State $p_{j \bmod t}$ exists if (1), (2), and (3) hold, which are equivalent to that all the three literals in I are true. Thus, every clause is satisfied in I .

Conversely, suppose that there exists a satisfying assignment, and let $\ell \leq p_1 \cdot \dots \cdot p_{|\mathcal{V}|}$ be the smallest integer such that $p_i \mid \ell$ if and only if v_i is true in the assignment. Then $S \cdot (a^{-1})^\ell$ has size $2|\mathcal{C}|$, since every state $p_{\ell \bmod t}$ exists. \square

Proposition 8. *When the automaton is unary, Problem 2 and Problem 4 are solvable in LOGSPACE or in $O(n)$ time.*

Proof. To solve the problems in logarithmic space, it is enough to find the smallest $i \leq n - 1$ such that $S \cdot (a^{-1})^i = Q$. Verifying $S \cdot (a^{-1})^i = Q$ can be done in logarithmic space by checking for every state q whether $qa^i \in S$.

Alternatively, we can solve the problems in $O(n)$ time, by determining the states lying on a cycle, and checking whether all of them are in S . Then the shortest word extending S is the longest distance from a state to a state in a cycle. \square

3. AVOIDING A STATE

A word w *avoids* a state $q \in Q$ if $q \notin Qw$. A state is *avoidable* if there exists a word that avoids it. Avoiding is a particular case of extending:

Remark 9. A state q is avoidable by a word w if and only if $Q \setminus \{q\}$ is extensible by w .

Proof. This follows since $Q \cdot w \subseteq Q \setminus \{q\}$ if and only if $(Q \setminus \{q\}) \cdot w^{-1} = Q$. \square \square

Problem 5. Given $\mathcal{A} = (Q, \Sigma, \delta)$ and a state $q \in Q$, is q avoidable?

We provide a complete characterization of states that are avoidable:

Theorem 10. Let $\mathcal{A} = (Q, \Sigma, \delta)$ be a strongly connected automaton. For every $q \in Q$, state q is avoidable if and only if there exists $p \in Q \setminus \{q\}$ and $w \in \Sigma^*$ such that $qw = pw$.

Proof. Let p and w be the state and the word from the theorem for a given state q . Since the automaton is strongly connected, there is a word w' such that $(pw)w' = (qw)w' = p$. For each subset $S \subseteq Q$ such that $p \in S$ we have $p \in Sww'$. Moreover, if $q \in S$ then $|Sww'| < |S|$, because $\{q, p\}ww' = \{p\}$. If q is not avoidable, then all $Q(ww'), Q(ww')^2, \dots$ contains q and forms an infinite sequence of subsets of decreasing cardinality, which is a contradiction.

Now consider the other direction. Suppose for a contradiction that q is avoidable, but there is no state $p \in Q \setminus \{q\}$ such that $\{q, p\}$ can be compressed. Let u be a word of minimal rank in \mathcal{A} , and v be a word that avoids q . Then $w = uv$ has the same rank and also avoids q . Let \sim be the equivalence relation defined by

$$p_1 \sim p_2 \iff p_1w = p_2w.$$

The equivalence class $[p]_{\sim}$ of $p \in Q$ is $(pw)w^{-1}$. There are $|Q/\sim| = |Qw|$ equivalence classes and one of them is $\{q\}$, since q does not belong to a compressible pair of states. For every state $p \in Q$, we know that $|Qw \cap [p]_{\sim}| \leq 1$, because $[p]_{\sim}$ is compressed by w to a singleton and Qw cannot be compressed by any word. Note that every state $r \in Qw$ belongs to some class $[p]_{\sim}$. From the equality $|Q/\sim| = |Qw|$ we conclude that for every class $[p]_{\sim}$ there is a state $r \in Qw \cap [p]_{\sim}$, thus $|Qw \cap [p]_{\sim}| = 1$. In particular, $1 = |Qw \cap [q]_{\sim}| = |Qw \cap \{q\}|$. This contradicts that w avoids q . \square

Note that if \mathcal{A} is not strongly connected, then every state from a strongly connected component that is not a sink can be avoided. If a state belong to a sink component, then we can consider the sub-automaton of this sink component, and by Theorem 10 we know that, given $q \in Q$, it is sufficient to check whether q belongs to a compressible pairs of states. Hence, Problem 5 can be solved using the well known algorithm [13] computing the pair automaton, which performs a breadth-first search with inversed edges on the pairs of states. It works in $O(|\Sigma|n^2)$ time and $O(|\Sigma|n + n^2)$ space.

Finally, we state an open problem concerning the complexity class of Problem 5. It is easy to see that it is in NL, and also it can be shown that it is NL-complete in general case.

Open problem 1. Is Problem 5 NL-complete when the automaton is strongly connected?

3.1. Bounded length. We now turn our attention to determining the length required to avoid a state.

Problem 6. *Given \mathcal{A} , a state q and a number ℓ , is q avoidable with a word w of length at most ℓ ?*

Theorem 11. *Problem 6 is NP-complete, even if the automaton is simultaneously strongly connected, synchronizing, and binary.*

Proof. The problem is in NP, because we can non-deterministically guess a word w as a certificate, and verify $q \notin \delta(Q, w)$ in $O(kn)$ time. If the state q is avoidable, then the length of the shortest avoiding words is at most $O(n^2)$ [31]. Hence, the problem is solvable in non-deterministic quadratic time.

In order to prove that the problem is NP-hard, we present a polynomial time reduction from the problem of determining the reset threshold in a specific subclass of automata, which is known to be NP-complete.

Let us have an instance of this problem from the Eppstein's proof [13, Theorem 8]. Namely, for a given synchronizing automaton $\mathcal{B} = (Q_{\mathcal{B}}, \{0, 1\}, \delta_{\mathcal{B}})$ and an integer $m > 0$, we are to decide whether there is a reset word w of length at most m . We do not want to reproduce here the whole construction from the Eppstein proof but we need some ingredients of it. Specifically, \mathcal{B} is an automaton with a sink state $z \in Q_{\mathcal{B}}$, and there are two subsets $S = \{s_1, \dots, s_d\}$ and $F \subseteq Q_{\mathcal{B}}$ with the following properties:

- (1) Each state $q \in Q_{\mathcal{B}} \setminus S$ is reachable from a state $s \in S$ through a (directed) path in the underlying digraph of \mathcal{B} .
- (2) For each state $s \in S$ and each word w of length m , we have $\delta_{\mathcal{B}}(s, w) \in F \cup \{z\}$.
- (3) For each $f \in F$ we have $\delta_{\mathcal{B}}(f, 0) = \delta_{\mathcal{B}}(f, 1) = z$.
- (4) For each state $s \in S$ and each word w of length less than m , we have $\delta_{\mathcal{B}}(s, w) \notin F$.

In particular, it follows that each word of length $m + 1$ is reset. Deciding whether \mathcal{B} has a reset word of length m is NP-hard.

We transform the automaton \mathcal{B} into \mathcal{A}' by the following reduction. First, we add a subset $R = \{r_0, r_2, \dots, r_m\}$ of states to provide that z is not avoidable by words of length less than $m + 1$. The transitions of both letters 0, 1 are $\delta_{\mathcal{A}'}(r_i, 0) = \delta_{\mathcal{A}'}(r_i, 1) = r_{i+1}$ for $i = 0, \dots, m - 1$, and $\delta_{\mathcal{A}'}(r_m, 0) = \delta_{\mathcal{A}'}(r_m, 1) = z$.

Secondly, we add a set of states $S' = \{s'_1, \dots, s'_d\}$ of size $d = |S|$ and a letter 2 to make the automaton strongly connected. Letters 0, 1 map S' to the corresponding states from S , that is, $\delta_{\mathcal{A}'}(s'_i, 0) = \delta_{\mathcal{A}'}(s'_i, 1) = s_i \in S$. Letter 2 connects states $r_0, s'_1, s'_2, \dots, s'_d$ into one path, i.e.

$$\delta_{\mathcal{A}'}(r_0, 2) = s'_1, \quad \delta_{\mathcal{A}'}(s'_1, 2) = s'_2, \quad \dots, \quad \delta_{\mathcal{A}'}(s'_{d-1}, 2) = s'_d.$$

All the other transitions of 2 we define equal to the transitions of 0.

Finally, we transform \mathcal{A}' to the final automaton $\mathcal{A}(Q, \{a, b\}, \delta)$. We encode letters 0, 1, 2 by 3-letter words over $\{a, b\}$ alike it was done in [5].

Namely, for each state $q \in Q_{\mathcal{A}'} \setminus (F \cup \{z\})$, we add 3 new states q^a, q^b and define their transitions as follows:

$$\begin{aligned} \delta(q, a) &= q^a, & \delta(q^a, a) &= \delta_{\mathcal{A}'}(q, 0), & \delta(q^a, b) &= \delta_{\mathcal{A}'}(q, 2), \\ \delta(q, b) &= q^b, & \delta(q^b, a) &= \delta_{\mathcal{A}'}(q, 1), & \delta(q^b, b) &= \delta_{\mathcal{A}'}(q, 2). \end{aligned}$$

Then, aa corresponds to applying 0, ba corresponds to applying 1, and ab and bb correspond to applying 2. Denote this encoding function by ϕ , i.e. $\phi(0) = aa$, $\phi(1) = ba$, and $\phi(2) = ab$. We also

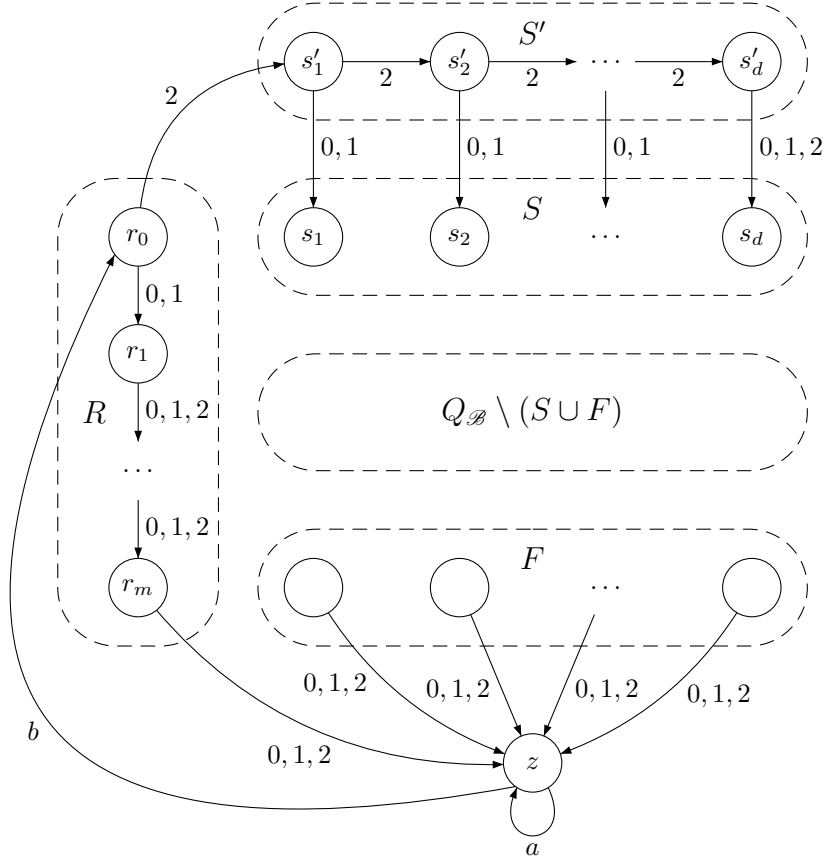


FIGURE 3. The automaton \mathcal{A} obtained from \mathcal{A}' in the proof of Theorem 11. Here every state q represents $\phi(q)$, and we have 0: aa , 1: ba , and 2: ab, bb .

extend ϕ to words over $\{0, 1, 2\}^*$ as usual. For simplicity, we denote also $\phi(q) = \{q, q^a, q^b\}$, and extend to subsets of $Q_{\mathcal{A}'}$ as usual.

It remains to define the transitions for $F \cup \{z\}$. We set $\delta(z, a) = z$, $\delta(z, b) = r_0$, and $\delta(f, a) = \delta(f, b) = z$ for each $f \in F$. Automaton \mathcal{A} is shown in Fig. 3.

Observe that \mathcal{A}' is strongly connected: z is reachable from each state, from z we can reach r_0 by 0, from r_0 we can reach every state from S' by applying a power of letter 2, and we can reach every state from S from the corresponding state from S' . Then every state from $Q_{\mathcal{B}}$ is reachable from a state from S . It follows that \mathcal{A} is also strongly connected, since for every $q \in Q_{\mathcal{A}'}$, every state from $\phi(q)$ is reachable from q .

Observe that \mathcal{A} is synchronizing: We claim that a^{2m+5} is a reset word for \mathcal{A}' . Indeed, aa does not map any state into $\phi(S')$. Every word of length $m+1$ is reset for \mathcal{B} and synchronizes to z , in particular, 0^{m+1} . Since $\phi(0^{m+1}) = a^{2m+2}$ does not contain bb , state z cannot go to S' by a factor of this word. Hence, we have

$$\delta(Q, a^{2m+4}) \subseteq \{z\} \cup \phi(R).$$

Then, finally, a^{m+1} compresses $\{z\} \cup \phi(R)$ to z .

Now, we claim that the original problem of checking whether \mathcal{B} has a reset word of length m is equivalent to determining whether z can be avoided in \mathcal{A} by a word of length at most $2m + 3$.

Suppose that \mathcal{B} has a reset word w of length m , and consider $u = \phi(0w)b$. Note that $\phi(0) = aa$ does not map any state into $\phi(S')$ nor into $\phi(r_0)$. Hence, we have

$$\delta(Q, \phi(0)) \subseteq \phi(Q_{\mathcal{B}}) \cup \phi(R \setminus \{r_0\}).$$

Due to the definition of ϕ , factor bb cannot appear in the image of words from $\{0, 1\}^*$ by ϕ . Henceforth, z cannot go to S' by a factor of $\phi(w)$. Since $|\phi(w)| = 2m$ and to map z into $\phi(r_m)$ we require a word of length $2m + 1$, the factors of $\phi(w)$ do not map z into $\phi(r_m)$. Since also w is a reset word for \mathcal{B} that maps every state from $Q_{\mathcal{B}}$ to z , we have

$$\delta(\phi(Q_{\mathcal{B}}), \phi(w)) \subseteq \{z\} \cup \phi(R \setminus \{r_m\}).$$

By the definition of the transitions on $R \cup \{z\}$ (only $\phi(2)$ maps r_0 outside), and since $|\phi(w)| = 2m$, we also have

$$\delta(\phi(R \setminus \{r_0\}), \phi(w)) \subseteq \{z\} \cup \phi(R \setminus \{r_m\}).$$

Finally, we get that $\delta(\{z\} \cup \phi(R \setminus \{r_m\}), b) \subset R$, thus u avoids z .

Let us prove the opposite direction. Suppose that state z can be avoided by a word u of length at most $2m + 3$. Then, by the definition of the transitions on R , the length of u must be exactly $2m + 3$ because $z \in \delta(R, w)$ for each w of length at most $2(m + 1)$. Let $u = u'u''u'''$ with $|u'| = 2$, $|u''| = 2m$, and $|u'''| = 1$. Notice that $S \subseteq \delta(S \cup S', u')$.

For words $w \in \{a, b\}^*$ of even length, we denote by $\tilde{\phi}^{-1}(w)$ the inverse image of encoding ϕ with respect to the definition on \mathcal{A}' , that is, $\tilde{\phi}^{-1}(aa) = 0$, $\tilde{\phi}^{-1}(ba) = 1$, $\tilde{\phi}^{-1}(ab) = \tilde{\phi}^{-1}(bb) = 2$, which is extended to words of even length by concatenation.

Remind that letter 2 acts like 0 on $Q_{\mathcal{B}}$ in \mathcal{A}' . By Property 2 of \mathcal{B} , every word of length m maps $s \in S$ into $\{z\} \cup F$. Then by Property 1 and 3, every state in $Q_{\mathcal{B}}$ is mapped to z . Hence, if the word $\tilde{\phi}^{-1}(u'')$ is not reset for \mathcal{B} , then there is a state $f \in F \cap \delta_{\mathcal{B}}(S, u'')$. This means that $f \in \delta(S, u'')$. But $\delta(f, a) = \delta(f, b) = z$, so u cannot avoid z . We conclude that the word $\tilde{\phi}^{-1}(u'')$ of length m is reset for \mathcal{B} . \square

Corollary 12. *Problem 4 is NP-complete when the automaton is synchronizing, and remains NP-hard when the automaton is simultaneously strongly connected, synchronizing, and binary.*

Proof. NP-hardness follows from Theorem 11, since we can set $S = Q \setminus \{q\}$. The problem is solvable in NP, because a suitable reset word extends a non-empty S to Q , and a shortest reset word is not longer than $O(n^3)$ [25, 31]. \square

Although much stronger inapproximability results was proven for the general case of automata of the following problem [14], all the proofs we could find deal only with non strongly connected automata. For completeness of the results, we state the following:

Corollary 13. *The problem of deciding whether there is a reset word of given length is NP-complete if the automaton is simultaneously strongly connected, synchronizing, and binary.*

Proof. In order to show this, let us consider slight modifications of the construction of \mathcal{A} from the proof of Theorem 11. All we have to do is to replace R with one state r and define transitions as follows:

$$\delta(z, b) = r, \quad r.b = s'_1, \quad r.a = z.$$

Clearly, \mathcal{A} remains strongly connected. So, it remains to show that it admits a reset word of length at most $2m + 2$ if and only if \mathcal{B} has a reset word of length m .

Let $w \in \{0, 1\}^*$ be a reset word for \mathcal{B} and denote $u = \phi(0w)$. Then, using that bb is not a factor of u and w is reset for \mathcal{B} , we get $\delta(Q, u) \subseteq \{z, r\}$. But r cannot be in the image of u because $r \notin \delta(Q, a)$ and u ends with a (since w ends with either 0 or 1). Thus, u of length $2|w| + 2$ is a reset word for \mathcal{A} .

Let us prove the opposite direction. Suppose that there is a reset word u of length $2m + 2$ for \mathcal{A} . Let $u = u'u''$ with $|u'| = 2$. Notice that $S \subseteq \delta(S \cup S', u')$. Recall that letter 2 acts like 0 on $Q_{\mathcal{B}}$ in \mathcal{A}' . By Property 2 of \mathcal{B} , every word of length m maps $s \in S$ into $F \cup \{z\}$. Suppose that the word $\tilde{\phi}^{-1}(u'')$ is not reset for \mathcal{B} , so there is a state $f \in F \cap \delta_{\mathcal{B}}(S, u'')$. It remains to notice that z cannot go to f by u : this is because if it is mapped to $s \in S$ through S' by some prefix of u (of length at least 4), then by Property 4 it cannot be mapped to F by the remaining suffix of u of length less than $2m$; hence, every prefix of u maps z into $\{r, z\}$. Therefore, u is not reset \mathcal{A} , which contradicts our assumption, so $\tilde{\phi}^{-1}(u'')$ is reset for \mathcal{B} . \square

3.2. Unary case.

Proposition 14. *When the automaton is unary, Problem 5 and Problem 6 are solvable in LOGSPACE or in $O(n)$.*

Proof. Observe that a state q is avoidable if and only if it does not lie on a cycle. This can be checked in logarithmic space or in $O(n)$ time by considering $q \cdot a^i$ for $i = 1, \dots, n$. If q does not lie on a cycle, the shortest avoiding word has length equal to the longest distance from a state p that can be mapped to q , which can be also checked in logarithmic space by enumerating states p or in $O(n)$ by reversing edges and a search from q . \square

4. CHANGING SIZE OF A SUBSET

In this section we deal with the following two problems:

Problem 7. *Given an automaton $\mathcal{A} = (Q, \Sigma, \delta)$ and a subset $S \subseteq Q$, is there a word $w \in \Sigma^*$ such that $|S \cdot w^{-1}| \neq |S|$?*

Problem 8. *Given an automaton $\mathcal{A} = (Q, \Sigma, \delta)$, a subset $S \subseteq Q$, and a number ℓ , is there a word $w \in \Sigma^*$ such that $|S \cdot w^{-1}| \neq |S|$ of length at most ℓ ?*

In contrast to the cases $|S \cdot w^{-1}| > |S|$ and $|S \cdot w^{-1}| < |S|$, there exists a polynomial time algorithm for both these problems.

For this purpose, we associate a natural linear structure with an automaton \mathcal{A} . By \mathbb{R}^n we denote the real n -dimensional linear space of row vectors. The value at an i -th entry of a vector $v \in \mathbb{R}^n$ we denote by $v(i)$. Without loss of generality, we assume that $Q = \{1, 2, \dots, n\}$ and then assign to each subset $K \subseteq Q$ its *characteristic vector* $[K] \in \mathbb{R}^n$, whose i -th entry $v(i) = 1$ if $i \in K$, and $v(i) = 0$, otherwise. For $q \in Q$ we write $[q]$ instead of $[\{q\}]$ to simplify the notation. By $\langle S \rangle$ we denote the linear span of $S \subseteq \mathbb{R}^n$. The *dimension* of a linear subspace L is denoted by $\dim(L)$. The $n \times n$ identity matrix is denoted by I_n .

Each word $w \in \Sigma^*$ corresponds to a linear transformation of \mathbb{R}^n . By $[w]$ we denote the matrix of this transformation in the standard basis $[1], \dots, [n]$ of \mathbb{R}^n . For example, if \mathcal{A} is the automaton from Fig. 1, then

$$[a] = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}, \quad [b] = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad [ba] = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

Clearly, as the automaton is deterministic, the matrix $[w]$ has exactly one non-zero entry in each row. In particular, $[w]$ is *row stochastic*, meaning that the sum of entries in each row is equal to 1.

In virtue of row-vector notation, we get that $[uv] = [u][v]$ for every two words $u, v \in \Sigma^*$. By $[w]^T$ we denote the transpose of the matrix $[w]$. One easily verifies that $[S \cdot w^{-1}] = [S][w]^T$. We extend the notation to sets of words W and by $[W]$ denote the set of matrices. Then, given a matrix $[w]$, by $[W][w]$ we mean $\{[w'][w] \mid w' \in W\}$. The same convention is used for $[W]v$, where $v \in \mathbb{R}^n$ is a vector, and $[W][W']$, where W' is another set of matrices; note that $[W][W'] = [WW']$. For two vectors $v_1, v_2 \in \mathbb{R}^n$, we denote their usual inner (scalar) product by (v_1, v_2) . By $\Sigma^{\leq c}$ we denote the set of all words over Σ of length at most c . The empty word is denoted by ε .

Now we are going to design a polynomial time algorithm solving Problem 7.

4.1. The filter function. First, consider the auxiliary FILTER function shown in Alg. 1. Its goal is to check whether a given vector $g \in \mathbb{R}^n$ is independent to the previously added vectors, and if so, include g in the internal set of vectors. A call to FILTER function with a given vector g returns true if and only if $G \cup \{g\}$ is an independent set of vectors, or equivalently $g \notin \langle G \rangle$.

To perform this subroutine efficiently, we maintain a sequence of vectors G and a sequence of indices I , which are empty at the beginning. Every time we use the Gaussian approach to reduce the matrix of vectors from G to a *pseudo-triangular* form. The sequence of (column) indices $I = (i_1, i_2, \dots, i_k)$ and normalized vectors $G = \{g'_1, \dots, g'_k\}$ have the property that for each j , $1 \leq j \leq k$, there is exactly one vector from $\{g'_1, \dots, g'_j\}$ with non-zero j -th coordinate, which is equal to 1.

We begin with the first non-zero vector g_1 , which can be normalized (by multiplying by a scalar) to the vector g'_1 having the coordinate with index i_1 equal to 1. Now, suppose we are given a vector g and we have already built G of size k and the set of indices $I = \{i_1, i_2, \dots, i_k\}$ with aforementioned properties. Then, we just compute $g' = g - \sum_{r=1}^k g(i_r) \cdot g_r$ (line 4). Since $g(i_r) \cdot g_r$ has $g(i_r)$ at the i_r -th coordinate, all the entries at the coordinates from I in g' are zero. If there is a non-zero coordinate left in g' , then take the first such coordinate (line 6), normalize g' to be 1 at this index and add to G (line 8). In the opposite case, if $g' = 0$, then g belongs to $\langle G \rangle$ and thus should not be added.

Assuming that in our computational model every arithmetic operation has a unitary cost, then clearly this function can be performed in $O(kn)$ -time during a k -th call. However, note that, if an exact computation is performed using rational numbers, then we may require to handle values of exponential order, and the total complexity would depend on the algorithms used for particular arithmetic operations.

4.2. The reduction procedure. Another auxiliary algorithm is the reduction procedure from [7] modified slightly to achieve the desired time complexity. The procedure is shown in Algorithm 2. It takes an integer $0 < d \leq n$ and a vector $\alpha \in \mathbb{R}^n$. Then it builds a set of words $W \subseteq \Sigma^{\leq d}$ of size equal to $\dim(\langle [\Sigma^{\leq d}] \alpha \rangle) \leq n$ such that $\langle [W] \alpha \rangle = \langle [\Sigma^{\leq d}] \alpha \rangle$ and that the maximum length of words from W is the smallest possible.

We iteratively feed vectors of the form $[w]\alpha$ to the FILTER function, keeping R – the set of vectors which were added together with the corresponding words. We start with only one pair (α, ε) . In each of at most d iterations, we construct a new set R' . Given a set of R pairs from the previous iteration, we consider all the vectors $\{[a]g \mid (g, w) \in R, a \in \Sigma\}$. We feed FILTER function with each of these vectors, and keep in R' these vectors that are independent to all the considered vectors so far (during all iterations). If nothing is added (R' is empty), we end the iterations; otherwise we

Algorithm 1 The filter function

```

1:  $G \leftarrow ()$  ▷ Empty sequence of vectors
2:  $I \leftarrow ()$  ▷ Empty sequence of indices
Require:  $g \in \mathbb{R}^n$ .
Ensure: Is  $g$  independent to all vectors from  $G$ .
3: function FILTER( $g$ )
4:    $g' \leftarrow g - \sum_{r=1}^{|I|} g(I[r]) \cdot G[r]$  ▷  $I[r]$  and  $G[r]$  denote the  $r$ -th element
5:   if  $g' \neq 0$  then
6:      $I \leftarrow I + \min\{i \mid g'(i) \neq 0\}$  ▷ Append index to  $I$ 
7:      $G \leftarrow G + g'/g'(j)$  ▷ Append vector to  $G$ 
8:     return true
9:   else
10:    return false
11:   end if
12: end function

```

Algorithm 2 The reduction procedure

Require: An automaton $\mathcal{A}_n(Q, \Sigma, \delta)$, a vector $\alpha \in \mathbb{R}^n$, and an integer $d \geq 1$.
Ensure: A set of words $W \subseteq \Sigma^{\leq k}$ such that $\langle W\alpha \rangle = \langle \Sigma^{\leq d}\alpha \rangle$, where $|W| = \dim(\langle \Sigma^{\leq d}\alpha \rangle)$ and $k \leq d$ is the smallest possible.

```

1:  $R \leftarrow ((\alpha, \varepsilon))$  ▷ Sequence of vector-number pairs
2:  $W \leftarrow \{\varepsilon\}$  ▷ Set of words
3: for  $i = 1, \dots, d$  do
4:    $R' \leftarrow \emptyset$ 
5:   for  $a \in \Sigma, (g, w) \in R$  do
6:     if FILTER( $g$ ) then
7:        $R' \leftarrow R' + (g, aw)$ 
8:     end if
9:   end for
10:  if  $R' = \emptyset$  then
11:    break
12:  else
13:     $R \leftarrow R'$ 
14:     $W \leftarrow \{w \mid \exists g \text{ such that } (g, w) \in R'\}$ 
15:  end if
16: end for
17: return  $W$ 

```

continue with the new $R = R'$. Simultaneously, we keep the set W of words w that were added to R' during some iteration.

Correctness. Now, let us prove the correctness of the reduction procedure. Let W_i be the set of words W after the i -th iteration, where also $W_0 = \{\varepsilon\}$. Let $V_i = \langle [W_i]\alpha \rangle$. Note that also $V_i = \langle G_i\alpha \rangle$, where G_i is the internal set of vectors in the FILTER function after the i -th iteration.

We set $V_{-1} = \{0\}$ (the empty subspace). Let R_i be the set of vectors $\{g \mid (g, w) \in R'\}$ that is added to R' in the i -th iteration, where also $R_0 = \{\alpha\}$. We prove by induction that $V_i = \langle [\Sigma^{\leq i}] \alpha \rangle$.

At the beginning, clearly $V_0 = \langle [\varepsilon] \alpha \rangle$. Let $i \geq 1$, and R be the set of vectors from the $(i-1)$ -th iteration. By the induction, we have

$$V_{i-1} = \langle [\Sigma^{\leq i-1}] \alpha \rangle = \langle \{[a]g \mid a \in \Sigma, g \in V_{i-2}\} \rangle = \langle R_{i-1} \cup V_{i-2} \rangle.$$

Therefore,

$$\langle [\Sigma^{\leq i}] \alpha \rangle = \langle \{[a]g \mid a \in \Sigma, g \in V_{i-1}\} \rangle = \langle \{[a]g \mid g \in R_{i-1}\} \cup \{[a]g \mid g \in V_{i-2}\} \rangle.$$

Since $\langle \{[a]g \mid g \in V_{i-2}\} \rangle$ is a subspace of V_{i-1} and $\langle \{[a]g \mid g \in R_{i-1}\} \rangle = \langle R_i \rangle$, the induction follows.

It follows from the ascending chain argument (see e.g. [19, 28]) that for some $j \leq n-1$ we have

$$V_0 \subsetneq V_1 \dots V_j = V_{j+1} = \dots$$

Thus, the procedure is stopped at j , and $j \leq \min\{d, n-1\}$. We get that $V_j = \langle [\Sigma^{\leq d}] \alpha \rangle$. Since in each iteration we add to W only words corresponding to independent vectors, we get that $|W| = \dim(V_j) \leq n$. Also, the lengths of words in W are at most $j \leq \min\{d, n-1\}$, and clearly j is equal to the length of the longest words from W , and since $\dim(\langle [\Sigma^{\leq j-1}] \alpha \rangle) < \dim(\langle [\Sigma^{\leq j}] \alpha \rangle)$, W must contain a word of length j . \square

Proposition 15. Denote $m = \dim(\langle [\Sigma^{\leq n}] \alpha \rangle)$. The time complexity of the reduction procedure is $O(nm^2|\Sigma|)$ and its space complexity is $O(nm)$.

Proof. In each iteration we run FILTER subroutine only for newly added vectors, that is, for each basis vector of $\langle [\Sigma^{\leq n}] [Q] \rangle$ we run it at most once for each letter. Hence, we run FILTER subroutine at most $|\Sigma| \cdot m$ times. Since the complexity of FILTER subroutine is $O(nm)$, these calls take $O(nm^2|\Sigma|)$ time.

Note that the sum of lengths of words in W is bounded by $O(mn)$. Hence, the other operations in the reduction procedure take at most $O(nm|\Sigma|)$ time.

The space complexity bound holds since the space used by W , G , R (R') is bounded by $O(mn)$. \square

4.3. Finding a shortest word w with $|S \cdot w^{-1}| \neq |S|$. Finally, we run the reduction procedure from Algorithm 2 with $\alpha = Q$, $d = n$, and additionally check in line 7 for each added vector g whether $([S], g) = |S|$. If the equality does not hold, then we return the corresponding word aw as the answer. If the procedure completes without facing this case, we answer that there is no such a word.

Theorem 16. Problem 7 can be solved in $O(n^3|\Sigma|)$ time and $O(n^2 + n|\Sigma|)$ space. Within the same complexities we can find a shortest word w with $|S \cdot w^{-1}| \neq |S|$, and hence solve Problem 8.

Proof. The correctness of our algorithm follows from the equation

$$(4) \quad |S \cdot w^{-1}| = ([S \cdot w^{-1}], [Q]) = ([S], [w][Q]).$$

Indeed, if w is a shortest word with the required property, then due to the reduction procedure properties, at the $|w|$ -th iteration $[w][Q] \in \langle [\Sigma^{\leq |w|}] [Q] \rangle = V_{|w|}$. Hence $[w][Q] = \sum_{u \in W_{|w|}} \lambda_u [u][Q]$ for some coefficients λ_u . Since for every word u we have $([Q], [u][Q]) = ([Q], [w][Q]) = ([Q], [Q]) = 1$, the sum of the coefficients λ_u is equal to 1. Therefore, if for each $u \in W_{|w|}$ we have $([S], [u][Q]) = |S|$, then

$$([S], [w][Q]) = \sum_{u \in W_{|w|}} \lambda_u ([S], [u][Q]) = |S|,$$

which contradicts our assumption about w .

Thus, by linearity, in the $|w|$ -th iteration there is at least one added vector g such that $([S], g) \neq [S]$. Then we have the corresponding word $u \in \Sigma^{\leq |w|}$ such that $g = [u][Q]$, which satisfies the desired property.

If there are no such words, the reduction procedure certainly will not stop with a word returned, and the theorem follows.

Since $m = \dim(\langle [\Sigma^{\leq n}][Q] \rangle) \leq n$, the complexity follows from Proposition 15. \square

We note that Problem 7 becomes trivial when the automaton is synchronizing: a word changing size exists if and only if $S \neq \emptyset$ and $S \neq Q$, since every such S is extensible to Q .

The running time $O(n^3|\Sigma|)$ of the algorithm is quite large (and may require a large arithmetic as discussed in the filter function). It remains open whether the problems can be solved faster.

Open problem 2. *Is there a faster algorithm for Problems 7 and 8?*

4.4. Unary case.

Proposition 17. *When the automaton is unary, Problem 7 and Problem 8 are solvable in LOGSPACE or in $O(n^2)$ time.*

Proof. By the proof of Theorem 16, the shortest word with the required property is of length at most $n - 1$, so it is enough to consider subsets $S \cdot a^{-1}, S \cdot (a^2)^{-1}, \dots, S \cdot (a^{n-1})^{-1}$.

It follows that we can check the problem in LOGSPACE by counting the cardinalities of these preimages. Alternatively, we can compute the $n - 1$ preimages directly, where each next preimage is computed in $O(n)$ time. \square

Open problem 3. *In the unary case, is there a faster algorithm for Problem 7 or Problem 8 than $O(n^2)$?*

REFERENCES

- [1] D. S. Ananichev and V. V. Gusev. Approximation of Reset Thresholds with Greedy Algorithms. *Fundamenta Informaticae*, 145(3):221–227, 2016.
- [2] D. S. Ananichev and M. V. Volkov. Synchronizing generalized monotonic automata. *Theoretical Computer Science*, 330(1):3–13, 2005.
- [3] M.-P. Béal, M. Berlinkov, and D. Perrin. A quadratic upper bound on the size of a synchronizing word in one-cluster automata. *International Journal of Foundations of Computer Science*, 22(2):277–288, 2011.
- [4] M. Berlinkov. Synchronizing Quasi-Eulerian and Quasi-one-cluster Automata. *International Journal of Foundations of Computer Science*, 24(6):729–745, 2013.
- [5] M. Berlinkov. On Two Algorithmic Problems about Synchronizing Automata. In *Developments in Language Theory*, LNCS, pages 61–67. Springer, 2014.
- [6] M. Berlinkov. On the probability of being synchronizable. In *CALDAM*, volume 9602 of *LNCS*, pages 73–84. Springer, 2016.
- [7] M. Berlinkov and M. Szykuła. Algebraic Synchronization Criterion and Computing Reset Words. In *Mathematical Foundations of Computer Science*, volume 9234 of *LNCS*, pages 103–115. Springer, 2015.
- [8] M. Berlinkov and M. Szykuła. Synchronizing Automata with Extremal Properties. In *Mathematical Foundations of Computer Science*, volume 9234 of *LNCS*, pages 331–343. Springer, 2015.
- [9] M. Berlinkov and M. Szykuła. Algebraic synchronization criterion and computing reset words. *Information Sciences*, 369:718–730, 2016.
- [10] M. T. Biskup and W. Plandowski. Shortest synchronizing strings for Huffman codes. *Theoretical Computer Science*, 410(38-40):3925–3941, 2009.
- [11] E. A. Bondar and M. V. Volkov. Completely reachable automata. In C. Câmpeanu, F. Manea, and J. Shallit, editors, *DCFS*, LNCS, pages 1–17. Springer, 2016.

- [12] J. Černý. Poznámka k homogénnym experimentom s konečnými automatami. *Matematicko-fyzikálny Časopis Slovenskej Akadémie Vied*, 14(3):208–216, 1964. In Slovak.
- [13] D. Eppstein. Reset sequences for monotonic automata. *SIAM Journal on Computing*, 19:500–510, 1990.
- [14] P. Gawrychowski and D. Straszak. Strong inapproximability of the shortest reset word. In *Mathematical Foundations of Computer Science*, volume 9234 of *LNCS*, pages 243–255. Springer, 2015.
- [15] F. Gonze, R. M. Jungers, and A. N. Trahtman. A Note on a Recent Attempt to Improve the Pin-Frankl Bound. *Discrete Mathematics and Theoretical Computer Science*, 17(1):307–308, 2015.
- [16] M. Grech and A. Kisielewicz. The Černý conjecture for automata respecting intervals of a directed graph. *Discrete Mathematics and Theoretical Computer Science*, 15(3):61–72, 2013.
- [17] K. Guldstrand Larsen, S. Laursen, and J. Srba. Synchronizing Strategies under Partial Observability. In P. Baldan and D. Gorla, editors, *CONCUR 2014*, pages 188–202. Springer, 2014.
- [18] H. Jürgensen. Synchronization. *Information and Computation*, 206(9-10):1033–1044, 2008.
- [19] J. Kari. Synchronizing finite automata on Eulerian digraphs. *Theoretical Computer Science*, 295(1-3):223–232, 2003.
- [20] J. Kari and M. V. Volkov. Černý’s conjecture and the road coloring problem. In *Handbook of Automata*. European Science Foundation, 2013.
- [21] A. Kisielewicz, J. Kowalski, and M. Szykuła. Computing the shortest reset words of synchronizing automata. *Journal of Combinatorial Optimization*, 29(1):88–124, 2015.
- [22] D. Kozen. Lower Bounds for Natural Proof Systems. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, SFCS’77, pages 254–266, 1977.
- [23] P. Martyugin. Computational Complexity of Certain Problems Related to Carefully Synchronizing Words for Partial Automata and Directing Words for Nondeterministic Automata. *Theory of Computing Systems*, 54(2):293–304, 2014.
- [24] J. Olschewski and M. Ummels. The complexity of finding reset words in finite automata. In *Mathematical Foundations of Computer Science*, volume 6281 of *LNCS*, pages 568–579. Springer, 2010.
- [25] J.-E. Pin. On two combinatorial problems arising from automata theory. In *Proceedings of the International Colloquium on Graph Theory and Combinatorics*, volume 75 of *North-Holland Mathematics Studies*, pages 535–548, 1983.
- [26] I. K. Rystsov. Polynomial complete problems in automata theory. *Information Processing Letters*, 16(3):147–151, 1983.
- [27] S. Sandberg. Homing and synchronizing sequences. In *Model-Based Testing of Reactive Systems*, volume 3472 of *LNCS*, pages 5–33. Springer, 2005.
- [28] B. Steinberg. The averaging trick and the Černý conjecture. *International Journal of Foundations of Computer Science*, 22(7):1697–1706, 2011.
- [29] B. Steinberg. The Černý conjecture for one-cluster automata with prime length cycle. *Theoretical Computer Science*, 412(39):5487–5491, 2011.
- [30] L. J. Stockmeyer and A. R. Meyer. Word Problems Requiring Exponential Time. In *Proceedings of the Fifth Annual ACM Symposium on Theory of Computing*, STOC, pages 1–9, 1973.
- [31] M. Szykuła. Improving the upper bound the length of the shortest reset words. <http://arxiv.org/abs/1702.05455>, 2017.
- [32] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.
- [33] A. N. Trahtman. The Černý conjecture for aperiodic automata. *Discrete Mathematics and Theoretical Computer Science*, 9(2):3–10, 2007.
- [34] A. N. Trahtman. Modifying the upper bound on the length of minimal synchronizing word. In *Fundamentals of Computation Theory*, volume 6914 of *LNCS*, pages 173–180. Springer, 2011.
- [35] M. V. Volkov. Synchronizing automata and the Černý conjecture. In *Language and Automata Theory and Applications*, volume 5196 of *LNCS*, pages 11–27. Springer, 2008.
- [36] M. V. Volkov. Synchronizing automata preserving a chain of partial orders. *Theoretical Computer Science*, 410(37):3513–3519, 2009.
- [37] V. Vorel. Subset Synchronization of Transitive Automata. In *Proceedings 14th International Conference on Automata and Formal Languages (AFL 2014)*, pages 370–381, 2014.